Designing Installations for Verification of the Model of Active Queue Management Discipline RED in the GNS3

T. R. Velieva,[1,*] A. V. Korolkova,[1,†] and D. S. Kulyabov[1,2,‡]

[1] *Department of Applied Probability and Informatics*
Peoples' Friendship University of Russia
Miklukho-Maklaya str. 6, Moscow, 117198, Russia
[2] *Laboratory of Information Technologies*
Joint Institute for Nuclear Research
Joliot-Curie 6, Dubna, Moscow region, 141980, Russia[§]

The problem of RED-module mathematical model results verification, based on GNS3 experimental stand, is discussed in this article. The experimental stand consists of virtual Cisco router, traffic generator D-ITG and traffic receiver. The process of construction of such stand is presented. Also, the interaction between experimental stand and a computer of investigation in order to obtain and analyze data from stand is revised.

I. INTRODUCTION

A stochastic model of the traffic management RED type module was built [1–4]. Verification of the model was carried out on the NS-2 basis. However, we would like to conduct verification on a real router. As a result was the task of designing an experimental stand. It was decided to verify the clean RED algorithm [5] based on Cisco router. For the construction of the stand software package GNS3 (Graphical Network Simulator) was chosen.

Thus, the purpose of the study is to build on the GNS3 basis a virtual stand consisting of a Cisco router, a traffic generator and a receiver. A traffic generator D-ITG (Distributed Internet Traffic Generator) is used as.

II. THE MODEL OF THE ACTIVE QUEUE MANAGEMENT RED MODULE

The model of the active queue management RED module is a development of fluid model [1–3, 6, 7]. In the works [8–10] for methodology unification the method of one-step process randomization was used. Based on one-step processes we construct the stochastic model of RED, and the model contains two main elements: source of traffic and receiver.

The source sends packets, the receiver processes the packets and send an acknowledge. Our model is based on the assumption that the source and receiver interact with management. Thus, we obtain two equations, one of which describes a TCP window, and the second — instant queue length. The intensity of sending packets depends on the window size.

The detailed description of RED stochastic model is given in [4] and in appendix (section), where was obtained:

$$\begin{cases} \mathrm{d}W = \dfrac{1}{T}\mathrm{d}t - \dfrac{W}{2}\mathrm{d}N + \sqrt{\dfrac{1}{T} + \dfrac{W}{2}\dfrac{\mathrm{d}N}{\mathrm{d}t}}\mathrm{d}V^1, \\ \mathrm{d}Q = \left(\dfrac{W}{T} - C\right)\mathrm{d}Q + \sqrt{\dfrac{W}{T} - C}\mathrm{d}V^2, \\ \dfrac{\mathrm{d}\hat{Q}}{\mathrm{d}t} = w_q C(Q - \hat{Q}). \end{cases}$$

Here W is the window size average value, Q is the average value of the instantaneous queue, $\hat{Q}$ is exponentially weighted moving average, C is service intensity, $\mathrm{d}V^i$ is two dimensional Wiener random process.

III. CISCO IOS STRUCTURE

To use the IOS image one should understand, which features given delivery option support [11]. The parameter "Feature set" (fig. 1) is responsible for this:

- IP Base: initial level of functionality is included in all other sets of features. Provides a basic routing, that is static routes, RIP, OSPF, EIGRP, only IPv4. Includes VLAN (802.1Q and ISL), which previously were available only in the IP Plus set. Also includes NAT.
- IP Services (3rd level switches): dynamic routing protocols, NAT, IP SLA.
- Advanced IP Services: support IPv6 is added.
- IP Voice: functionality VoIP and VoFR is added.
- Advanced Security: IOS/Firewall, IDS, SCTP, SSH and IPSec (DES, 3DES and AES) are added.

* trvelieva@gmail.com
† akorolkova@sci.pfu.edu.ru
‡ yamadharma@gmail.com
§ Published in: T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, Designing installations for verification of the model of active queue management discipline RED in the GNS3, in: 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, 2014, pp. 570–577. doi:10.1109/ICUMT.2014.7002164. ; Sources: `https://bitbucket.org/yamadharma/articles-2014-gns3-red`

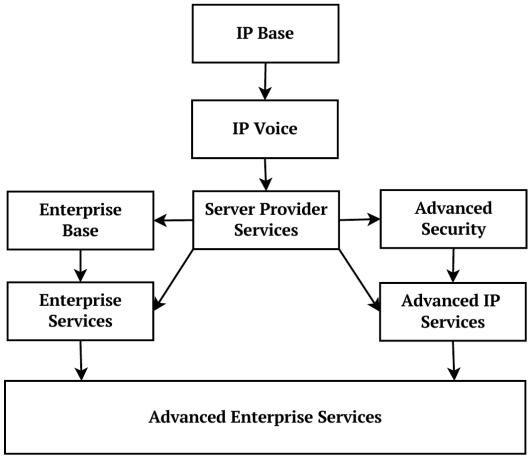

Figure 1. Feature tree of the new switch IOS naming convention

Table I. New naming convention for Cisco Router IOS

Code	Features
Base	Entry level image (IP Base, Enterprise Base)
Services	Addition of IP Telephony Service, MPLS, Voice over IP, Voice over Frame Relay and ATM (Included in SP Services, Enterprise Services)
Advanced	Addition of VPN, Cisco IOS Firewall, 3DES encryption, SSH, Cisco IOS IPSec and Intrusion Detection Systems (IDS) (Advanced Security, Advanced IP Services)
Enterprise	Addition of multi-protocols, including IBM, IPX, AppleTalk (Enterprise Base, Enterprise Services)

- Service Provider Services: IPv6, Netflow, SSH, BGP, ATM and VoATM are added.
- Enterprise Base: the support for L3 protocols such as IPX and AppleTalk is added. Also IBM DLSw+, STUN/BSTUN and RSRB are added.

Since version IOS 12.3T Cisco uses a new naming scheme for images (table I). However, this method of naming does not cover all the subtleties of image acquisition, so elements of the old naming scheme(table II) are still used.

For unknown reasons, not all IOS images are efficient in GNS3. Several images were tested. Here's a short list of working and non-working images.

Workable images:

- `C1700-adventerprisek9-mz.124-8.`
- `C1710-bk9no3r2sy-mz.124-23`
- `C1720-l2sy-mz.121-11`
- `C2600-adventerprisek9_sna-mz.124-25b`

Table II. Old naming convention for Cisco Router IOS

Code	Features
I	IP
Y	IP on 1700 Series Platforms
S	IP Plus
S6	IP Plus – No ATM
S7	IP Plus – No Voice
J	Enterprise
O	IOS Firewall/Intrusion Detection
K	Cryptorgaphy/IPSEC/SSH
K8	56Bit DES Encryption (Weak Cryptography)
K9	3DES/AES Encryption (Strong Cryptography)
X	H323
G	Services Selection Gateway (SSG)
C	Remote Access Server or Packet Data Serving Node (PDSN)
B	Apple Talk
N	Novel IP/IPX
V	Vox
R	IBM
U	Unlawful Intercept
P	Service Provider
Telco	Telecommunications Feature Set
Boot	Boot Image (Used on high end routers/switches)

- `C2691-adventerprisek9_sna-mz.124-23`
- `C3660-jsx-mz.123-4.T.`
- `C3745-adventerprisek9_sna-mz.124-15.T14.`
- `C7200-adventerprisek9_sna-mz.152-4.m4`

Unworkable:

- `C2600-adventerprisek9-sna-mz.124-23.`
- `C3745-adventerprisek9_sna-mz.124-15.T14`
- `C3745-adventerprisek9_ivs-mz.124-15.T14`
- `C3745-adventerprisek9_ivs-mz.124-15.T8`

IV. GNS3 INSTALLATION AND CONFIGURATION

GNS3 allows you to simulate a virtual network consisting of routers and virtual machines [12]. In fact, it is a graphical interface for different virtual machines. To emulate Cisco devices emulator dynamips is used. In addition, you can use such emulators as VirtualBox and Qemu. The latter is particularly useful when we work with the KVM, allowing the hardware implementation of the processor. The graphical interface makes it easy to switch different virtual machines. Also, there is the opportunity to connect projected topology with the real network. Wireshark allows to monitor traffic within the designed topology.

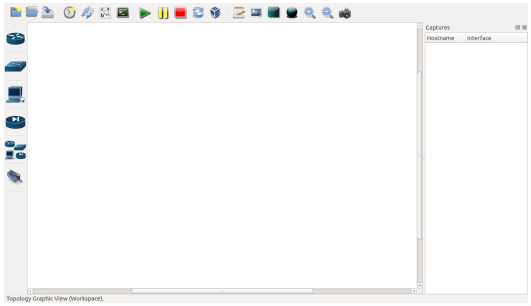

Figure 2. GNS3 interface

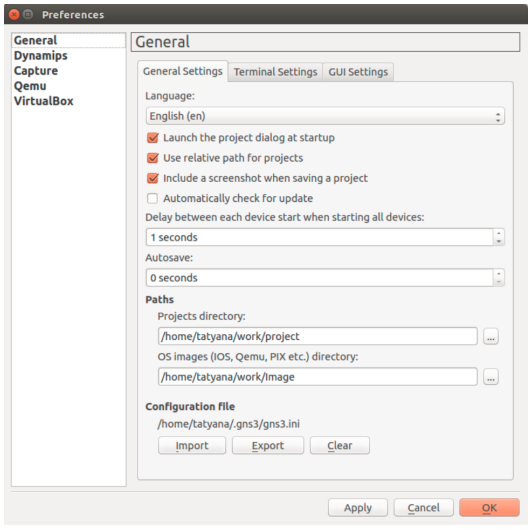

Figure 3. General preferences of GNS3

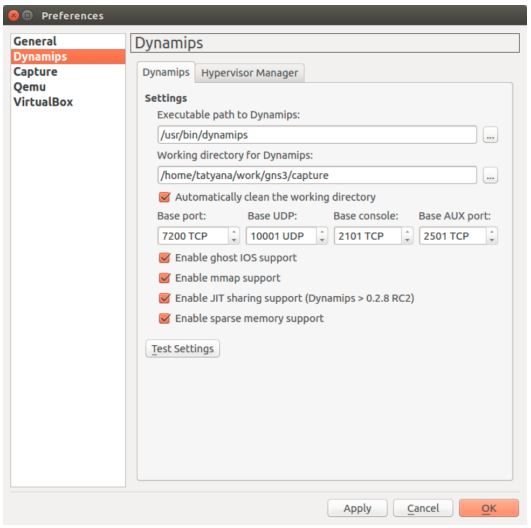

Figure 4. Dynamips preferences

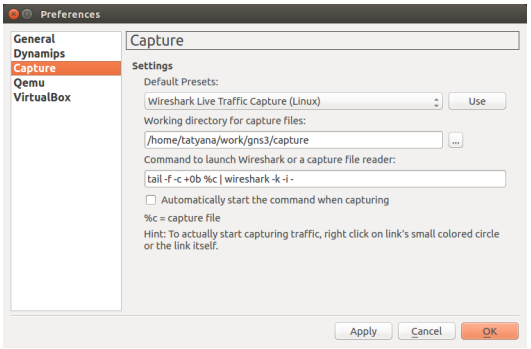

Figure 5. Capture preferences

To work with GNS3 we need to install Dynamips, VirtualBox and/or QEMU, xdotool, Wireshark. To install the above software for Linux operating systems (in our case, GNS3 installed on Ubuntu 14.04) the following commands are prescribed in the console:

```
sudo apt-get install dynamips
sudo apt-get install qemu
sudo apt-get install virtualbox
sudo apt-get install xdotool
sudo apt-get install wireshark
```

After that GNS3 is set with a command in a terminal:

```
sudo apt-get install gns3
```

Then GNS3 is run. GNS3 interface opens (fig. 2).

At the top the context menu, on the left-hand — the equipment to choose, at the bottom — a console window, on the right-hand — a network management menu. Before start you need to pre-configure GNS3.

For this the item Preferences is selected in the Edit of context menu. (fig. 3).

The language can be changed in General submenu. Here the path to the folder with designs and images of equipment is prescribed (fig. 4).

In submenu Capture traffic capture parameters are configured (fig. 5) (however, in this article we do not use the opportunity to capture the traffic). In line Default Presents the option Wireshark Live Traffic Capture (Linux) is choosed. In the next line, the path to the directory to store data capture is specified. And the last line the command to start Wireshark capture is proscribed:

```
tail -f -c +0b %c | wireshark -k -i
```

In the Qemu the tab General Settings sets the path to Qemuwrapper (this file is supplied with GNS3). The directory for capture is specified. In line Path to qemu the path to the file that emulates the processor is set. The following line specifies the path to the virtual machine. Port numbers are reserved by default. To test Qemuwrapper one need to click on the button Test Settings, a green label on the implementation of the test should appear (fig. 6).

Next we need to install the OS image to the virtual machine. Set of images can be taken from the page `http://www.gns3.net/appliances/`. We chose Linux Core 4.7.7, because it contains a traffic generator D-ITG.

In line Qemu flavor we specify the model and name of processor. After that the path to the OS image is set. Push Save and OK. After that, the Qemu guest can be

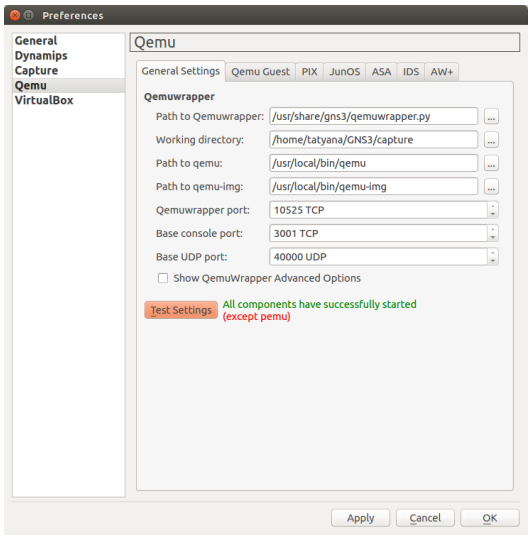

Figure 6. Preferences GNS3 - Qemu

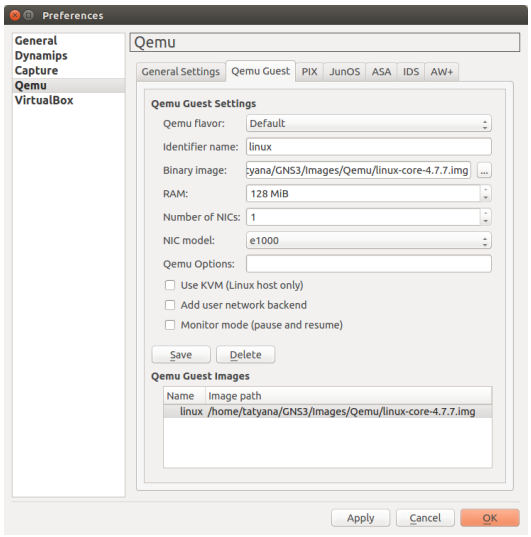

Figure 7. Preferences GNS3

selected on the toolbox (fig. 7).

Now you can assemble the stand. For this we portable from device selection menu router and virtual machine Qemu guest, rename (right mouse button › change the hostname) (fig. 8).

Host01 is a package source; host02 is a destination. Let's configure the router slots. To do this, right click on router01 and select Configure › router01 › Slots. For slot0 we select FastEthernet (fig. 9).

Then we create a connection between the router and the virtual machine. We should right-click on host01, select add link › FastEthernet › e0 and connect with router01, choosing f0/0. A connection between host02 and router01 is created similarly, only now we choose Ethernet type of connection (speed 10 Mbps).

Now we need to connect the virtual machines with the

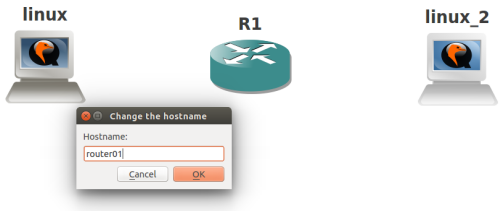

Figure 8. Virtual Network

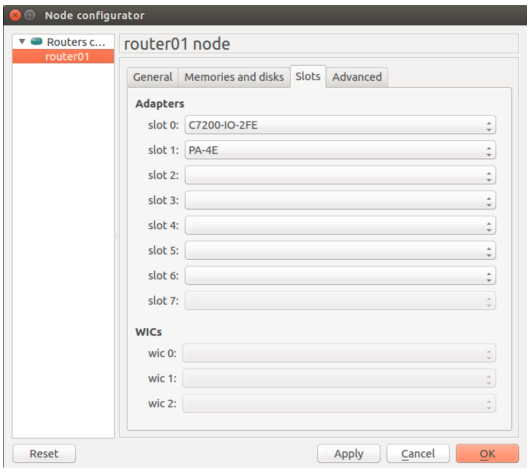

Figure 9. Router configurator

host computer (for receiving and processing the logs). Connection will be carried through TUN/ TAP interface (TAP simulates an Ethernet device and works at the link layer model OSI, Ethernet frames and terms used to create a bridge). To do this, drag it to the workspace cloud and configure it: Configure › NIO TAP. Write the name tap0 and press add › apply › ok (fig. 10).

We create a connection to router01, selecting the type of connection FastEthernet, and get topology (fig. 11).

Next, you need to configure the devices.

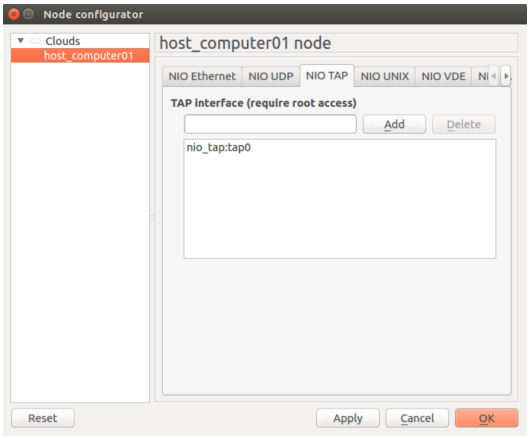

Figure 10. Cloud interface configurator

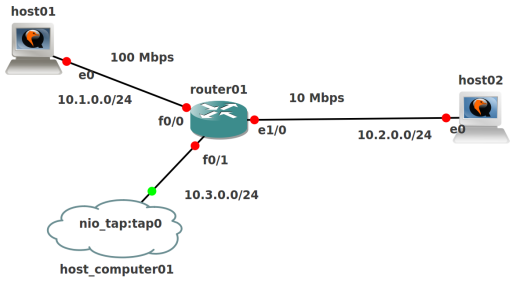

Figure 11. Virtual Network

```
router01
router01(config-if)#exit
router01(config)#interface e1/0
router01(config-if)#ip address 10.2.0.1 255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#full-duplex
router01(config-if)#exit
router01(config)#interface f0/1
router01(config-if)#ip address 10.3.0.1 255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#exit
router01#
*Jun 15 18:04:50.867: %SYS-5-CONFIG_I: Configured from console by console
router01#wr mem
Warning: Attempting to overwrite an NVRAM configuration previously written
by a different version of the system image.
Overwrite the previous NVRAM configuration?[confirm]
Building configuration...
[OK]
router01#/usr/bin/filetool.sh -b
                             ^
% Invalid input detected at '^' marker.

router01#/usr/bin/filetool.sh -b
```

Figure 12. Router CLI configuration

Run router01 by clicking (right button ⟩ start). Open the console for it (right button ⟩ Console) and enter the following commands (fig. 12):

```
Router>enable
Router#configure terminal
router01(config)#hostname router01
router01(config)#interface f0/0
router01(config-if)#ip address 10.1.0.1
  255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#interface e1/0
router01(config-if)#ip address 10.2.0.1
  255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#full-duplex
router01(config-if)#exit
router01(config)#interface f0/1
router01(config-if)#ip address 10.3.0.1
  255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#exit
router01#write memory
```

Now we open the console for host01. As the name of the user we specifie `tc`, the password is blank. Go to the superuser: `sudo su`. Edit parameters host01 (to start editing in vi we should enter `i`; to save the changes and exit we should press the button esc, then enter `:wq` and press the Enter). We need to edit two files: `/opt/bootsync.sh`

```
root@tatyana: ~
tatyana@tatyana:~$ sudo -i
[sudo] password for tatyana:
root@tatyana:~# tunctl -u tatyana -t tap0
Set 'tap0' persistent and owned by uid 1000
root@tatyana:~# ifconfig tap0 10.3.0.10 netmask 255.255.255.0 up
root@tatyana:~#
```

Figure 13. Address configuration on tap0 interface

```
root@tatyana: ~
tatyana@tatyana:~$ sudo -i
[sudo] password for tatyana:
root@tatyana:~# tunctl -u tatyana -t tap0
Set 'tap0' persistent and owned by uid 1000
root@tatyana:~# ifconfig tap0 10.3.0.10 netmask 255.255.255.0 up
root@tatyana:~# iptables -I INPUT 1 -i tap0 -j ACCEPT
root@tatyana:~# route add -net 10.1.0.0/24 dev tap0
root@tatyana:~# route add -net 10.2.0.0/24 dev tap0
root@tatyana:~#
```

Figure 14. Routing configuration on tap0 interface

and `/opt/bootlocal.sh`:

```
vi /opt/bootlocal.sh
```

At end of file we write the following:

```
ifconfig eth0 10.1.0.10 netmask 255.255.255.0 up
route add default gw 10.1.0.1
```

After saving the changes to a file we exit from it, and execute the command to save the configuration (Linux Core uses the Tinycore configuration system):

```
/usr/bin/filetool.sh -b
```

Similarly host02 is configured, but ip address (network 10.2.0.0/24) should be changed.

Let's configure the host computer with TAP-interface. In the command window as a root we create an interface for user:

```
tunctl -u user_name -t tap0
```

Now we set the address for tap0 interface (fig. 13):

```
ifconfig tap0 10.3.0.10 netmask 255.255.255.0 up
```

Then you can configure filtering and routing (fig. 14):

```
iptables -I INPUT 1 -i tap0 -j ACCEPT
route add -net 10.1.0.0/24 dev tap0
route add -net 10.2.0.0/24 dev tap0
```

V. TRAFFIC GENERATOR D-ITG

Now we can generate traffic and take readings. To generate traffic we use D-ITG.

D-ITG provides estimates for key indicators of quality of service (the average packet delay, delay variation (jitter), packet loss ratio, throughput) with a high degree of certainty. Depending on the requirements of the experiment, we can change the following parameters:

- number of streams transmitted between end stations;

- duration of traffic generation;
- the intensity of each separate stream (pack/s or bit/s);
- packet traffic length or distribution law;
- type and parameters of the distribution law of the time interval between adjacent packets (for example, packet length and time interval between packets can be distributed in a uniform, exponential, normal, gamma-law or Pareto, Cauchy, Poisson);
- type of transport layer protocols: TCP, UDP, DCCP, SCTP;
- type of traffic (simulation of a flow generated by specific application protocol): VoIP, IPTV, Telnet, DNS, game traffic (Counter Strike, Quake3).

Within the D-ITG the control of experiment is performed by using the command line, and the required set of parameters to generate traffic is given by calling the program `ITGSend` using keys.

Let's consider the example of generating traffic for UDP and TCP. In host02 console we run the command of channel audition:

```
ITGRecv
```

In host01 console we run host01 traffic generation:

```
ITGSend -a 10.2.0.10 -T UDP -C 10000 -c 500 \
  -t 20000 -x recv_log_file
```

Here we transfer the files to the receiver address by 10.2.0.10 protocol UDP, the transmission rate is 10,000 packets per second, the size of package is 500 bytes, the connection time is 20000 ms. All the information about send data is recorded on the receiving end in a file called `recv_log_files`. Once data transfer has been performed, we stop listening to host02 console (pressing Ctrl+C), then run the command in host02 console to decode the log file:

```
ITGDec recv_log_file
```

By this command a table with the values of the received stream is displayed (fig. 15).

We take similarly steps for TCP and get the following results (fig. 16).

For multicast traffic in host01 command window for file `send` we describe each flow of traffic. For example, as follows:

```
cat > send <<EOF
-a 10.2.0.10  -C 1000 -c 512 -T UDP
-a 10.2.0.10  -C 2000 -c 512 -T UDP
-a 10.2.0.10  -C 3000 -c 512 -T UDP
-a 10.2.0.10  -C 4000 -c 512 -T UDP
-a 10.2.0.10  -C 5000 -c 512 -T UDP
EOF
```

```
ITGDec version 2.8.1 (r1023)
Compile-time options: dccp bursty multiport
|------------------------------------------------------------
Flow number: 1
From 10.1.0.10:57573
To    10.2.0.10:8999
------------------------------------------------------------
Total time                =     19.998072 s
Total packets             =          9257
Minimum delay             =      0.141002 s
Maximum delay             =      0.153761 s
Average delay             =      0.146050 s
Average jitter            =      0.003195 s
Delay standard deviation  =      0.002774 s
Bytes received            =       4739584
Average bitrate           =   1896.016376 Kbit/s
Average packet rate       =    462.894623 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS   ******************
------------------------------------------------------------
Number of flows           =             1
Total time                =     19.998072 s
Total packets             =          9257
Minimum delay             =      0.141002 s
Maximum delay             =      0.153761 s
Average delay             =      0.146050 s
Average jitter            =      0.003195 s
Delay standard deviation  =      0.002774 s
Bytes received            =       4739584
Average bitrate           =   1896.016376 Kbit/s
Average packet rate       =    462.894623 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
tc@host02:~$
```

Figure 15. Input flow statistics. UDP

Now let's run the traffic flow from host01. The transmission traffic data from host01 to host02 is written in the files send.log and recv.log:

```
ITGSend send -l send.log -x recv.log
```

Let's display the report of flow transmission. To do this, we write the command console host02

```
ITGDec send.log
```

to decrypt the sent traffic (fig. 17) and command

```
ITGDec recv.log
```

to decrypt the received traffic.

Thus, we obtain data about each flow, as well as the value of the outgoing and incoming flows parameters.

VI. VISUALIZATION OF RESULTS

Charts can be constructed for the following parameters: delay, jitter, bitrate, packet loss.

Assume that we have 5 streams, which are transmitted over TCP. Flow rate 1 is 1000 pps, flow 2 is 2000 pps, stream 3 is 3000 pps, stream 4 is 4000 pps, stream 5 is 5000 pps. Packet sizes are 512 bytes and the same for all streams, and Connection Time is 20000 ms. For packet loss demonstration we will use UDP protocol (fig. 18).

With the help of `|ITGplot|`we make the graphs of bitrate, delay and jitter. For this we need to record the values obtained in individual files using the following commands:

```
ITGDec version 2.8.1 (r1023)
Compile-time options: dccp bursty multiport
\------------------------------------------------------------
Flow number: 1
From 10.1.0.10:34185
To    10.2.0.10:8999
------------------------------------------------------------
Total time                =     20.015369 s
Total packets             =          4742
Minimum delay             =      0.132785 s
Maximum delay             =      1.180837 s
Average delay             =      0.190162 s
Average jitter            =      0.005034 s
Delay standard deviation  =      0.137616 s
Bytes received            =       2371000
Average bitrate           =    947.671762 Kbit/s
Average packet rate       =    236.917940 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------

****************  TOTAL RESULTS    ******************

------------------------------------------------------------
Number of flows           =             1
Total time                =     20.015369 s
Total packets             =          4742
Minimum delay             =      0.132785 s
Maximum delay             =      1.180837 s
Average delay             =      0.190162 s
Average jitter            =      0.005034 s
Delay standard deviation  =      0.137616 s
Bytes received            =       2371000
Average bitrate           =    947.671762 Kbit/s
Average packet rate       =    236.917940 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
root@host02:~#
```

Figure 16. Input flow statistics. TCP

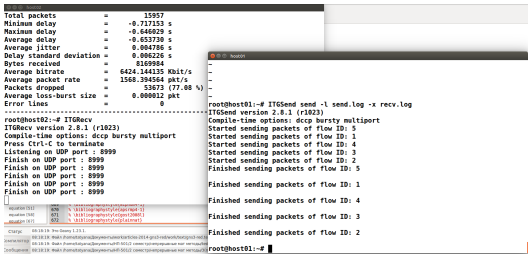

Figure 17. Log decryptiion

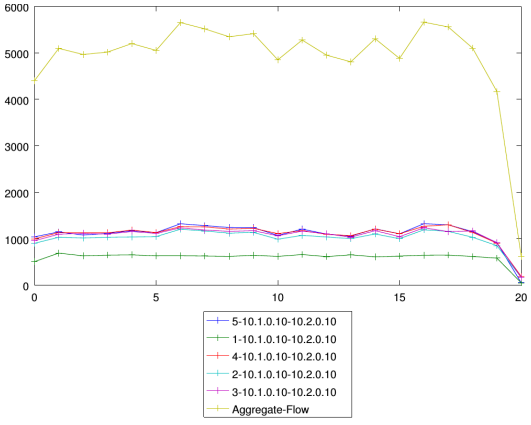

Figure 18. UDP packets loss

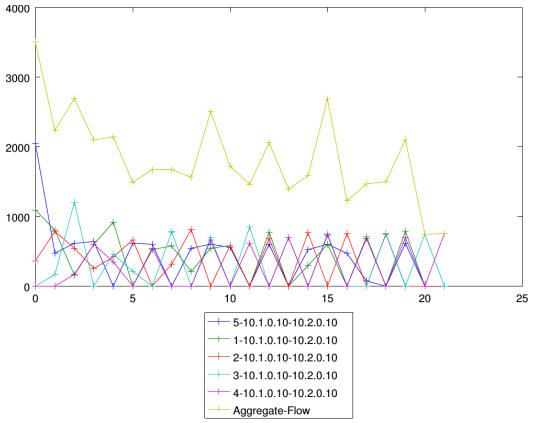

Figure 19. Incoming bitrate

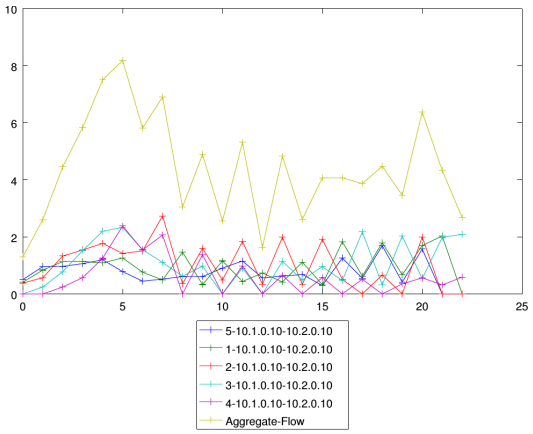

Figure 20. Incoming delay

```
ITGDec recv.log -b 1000
ITGDec recv.log -d 1000
ITGDec recv.log -j 1000
```

Every 1000 ms in files bitrate.dat, jitter.dat, delay.dat the respective values of the parameters are recorded for the transmitted flows (fig. 19, 20, 21). We construct graphs using the following commands:

```
./ITGplot birate.dat
./ITGplot delay.dat
./ITGplot jitter.dat
```

On obtained graphs the upper curve shows the general behavior of all flows.

VII. CONCLUSION

Thus, we have built the foundation of the stand for verification model of traffic management module RED. However, in this paper is not considered However, the task of RED module configuration on given router is not considered as well as a problem of real rime router reading.

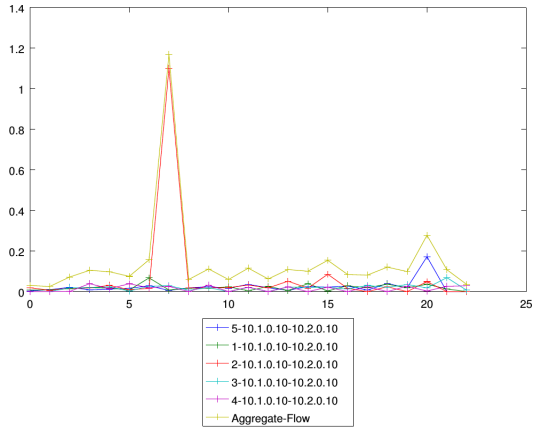

Figure 21. Incoming jitter

Appendix: Active queue management RED module stochastic model construction

The window change for each acknowledgement:

$$W_{n+1} = W_n + \frac{1}{W_n}.$$

For one-step processes stochastization we need to use the continuous time model. In round-trip time T there will be W_n acknowledgments:

$$\dot{W} = \frac{1}{T}.$$

The discrete equation of queue instantaneous length:

$$Q_{n+1} = Q_n + W_n - C_n.$$

Based on the discrete equation of queue instantaneous length the following equation is derived:

$$\dot{Q} = \frac{W(t)}{T(t)} - C(t).$$

Next, the behaviour of the exponentially weighted moving average queue length (the equation of connection between the source and the recipient) will be described.

The discrete recurrent equation of average queue length:

$$\hat{Q}(t_k + \delta) = (1 - w_q)\hat{Q}(t_k) + w_q Q(t_k).$$

From above mentioned discrete equation the continuous equation of average queue length is derived:

$$\dot{\hat{Q}} = -\frac{w_q}{\delta}\hat{Q} + \frac{w_q}{\delta}Q.$$

In order to consider the packets behavior in the system the kinetic equation, or as it is called the equation of interaction, will be presented. The number of packets is specified by the TCP window .

$$\begin{cases} 0 \xrightarrow{k_1^1} W, \\ W \xrightarrow{k_2^1} 0. \end{cases}$$

The first relation describes the appearance of packages in the system, the second describe departure of the packages from the system.

Based on the written equations and with the help of the method of constructing one-step processes the Fokker–Planck equation is derived:

$$\frac{\partial w}{\partial t} = -\frac{\partial}{\partial W}\left[\left(\frac{1}{W} - \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}\right)w\right] + \\ + \frac{1}{2}\frac{\partial^2}{\partial W^2}\left[\left(\frac{1}{W} + \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}\right)w\right].$$

From the Fokker–Planck equation the corresponding Langevin equation is obtained.

$$\mathrm{d}W = \frac{1}{W}\mathrm{d}t - \frac{W}{2}\mathrm{d}N + \sqrt{\frac{1}{W} + \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}}\mathrm{d}V^1,$$

where $\mathrm{d}V^1$ is the Wiener process corresponding to the random process $W(t)$.

Similarly, the behaviour of the queue length is described and interaction equations for the instantaneous queue length are given:

$$\begin{cases} 0 \xrightarrow{k_1^2} Q \\ 0 \xrightarrow{k_2^2} Q. \end{cases}$$

The Fokker–Planck equation for the instantaneous queue length:

$$\frac{\partial q}{\partial t} = -\frac{\partial}{\partial Q}\left[\left(\frac{W}{T} - C\right)q\right] + \frac{1}{2}\frac{\partial^2}{\partial Q^2}\left[\left(\frac{W}{T} - C\right)q\right].$$

The Langevin equation for the instantaneous queue length:

$$\mathrm{d}Q = \left(\frac{W}{T} - C\right)\mathrm{d}t + \sqrt{\frac{W}{T} - C}\mathrm{d}V^2,$$

where $\mathrm{d}V^2$ is the Wiener process which corresponds to the random process Q.

According to the obtained equations the resulting system of the equations is given:

$$\begin{cases} \mathrm{d}W = \frac{1}{T}\mathrm{d}t - \frac{W}{2}\mathrm{d}N + \sqrt{\frac{1}{T} + \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}}\mathrm{d}V^1, \\ \mathrm{d}Q = \left(\frac{W}{T} - C\right)\mathrm{d}Q + \sqrt{\frac{W}{T} - C}\mathrm{d}V^2, \\ \frac{\mathrm{d}\hat{Q}}{\mathrm{d}t} = w_q C(Q - \hat{Q}). \end{cases}$$

The detailed stochastic model of the router RED-like control module is described in [4].

[1] A. V. Korolkova, The Methods of Drop Probability Calculation for RED Algorithm, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (1–2) (2007) 32–37, in Russian.

[2] A. V. Korolkova, D. S. Kulyabov, A. I. Tchernoivanov, On the Classification of RED Algorithms, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (3) (2009) 34–46, in Russian.

[3] A. V. Korolkova, D. S. Kulyabov, Mathematical Model of the Dynamic Behavior of RED-Like System Parameters, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (1) (2010) 68–76, in Russian.

[4] T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, B. A. dos Santos, Model Queue Management on Routers, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" 2 (2014) 81–92, in Russian.

[5] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413. doi:10.1109/90.251892.

[6] V. Misra, W.-B. Gong, D. Towsley, Stochastic Differential Equation Modeling and Analysis of TCP-windowsize Behavior, Proceedings of IFIP WG 7.3 Performance 99.
URL http://dna-pubs.cs.columbia.edu/citation/paperfile/24/Misra99-TCP-Stochastic.pdf

[7] V. Misra, W.-B. Gong, D. Towsley, Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED, ACM SIGCOMM Computer Communication Review 30 (4) (2000) 151–160. doi:10.1145/347057.347421.

[8] A. V. Demidova, D. S. Kulyabov, Introduction of Self-Consistent Term in Stochastic Population Model Equation, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (3) (2012) 69–78, in Russian.

[9] A. V. Demidova, M. N. Gevorkyan, A. D. Egorov, D. S. Kulyabov, A. V. Korolkova, L. A. Sevastyanov, Influence of Stochastization on One-Step Models, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (1) (2014) 71–85, in Russian.

[10] E. G. Eferina, A. V. Korolkova, M. N. Gevorkyan, D. S. Kulyabov, L. A. Sevastyanov, One-Step Stochastic Processes Simulation Software Package, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (3) (2014) 46–59.

[11] Cisco Feature Navigator.
URL www.cisco.com/go/cfn

[12] C. Welsh, GNS3 network simulation guide, PACKT Publisher, 2013.
URL http://cds.cern.ch/record/1633716

Конструирование установки для верификации модели модуля активного управления трафиком RED в среде GNS3

Т. Р. Велиева,[1,*] А. В. Королькова,[1,†] and Д. С. Кулябов[1,2,‡]

[1] *Кафедра прикладной информатики и теории вероятностей, Российский университет дружбы народов, ул. Миклухо-Маклая, д.6, Москва, Россия, 117198*

[2] *Лаборатория информационных технологий, Объединённый институт ядерных исследований, ул. Жолио-Кюри 6, Дубна, Московская область, Россия, 141980*[§]

При построении стохастической модели RED возникла проблема проведения этапа верификации. В качестве системы верификации предложено использовать реализацию протокола RED компании Cisco. Установка собрана в симуляторе оборудования Cisco — системе GNS3.

I. ВВЕДЕНИЕ

В нашем коллективе была построена стохастическая модель модуля управления трафиком типа RED [1–4]. Верификация модели была проведена на основе средства имитационного моделирования NS-2. Однако нам хотелось бы провести верификацию на реальном маршрутизаторе. В результате возникла задача по конструированию экспериментального стенда. Для начала было решено верифицировать чистый алгоритм RED [5] на основе маршрутизатора Cisco. Для построения стенда был выбран программный комплекс GNS3 (Graphical Network Simulator).

Таким образом, целью исследования является построение на основе GNS3 виртуального стенда, состоящего из маршрутизатора Cisco, генератора трафика и получателя. В качестве генератора трафика используется D-ITG (Distributed Internet Traffic Generator).

II. СТОХАСТИЧЕСКАЯ МОДЕЛЬ МОДУЛЯ УПРАВЛЕНИЯ ТРАФИКОМ RED

Данная модель является развитием жидкостной модели модуля управления трафиком RED [1–3, 6, 7]. Для унификации методики построения модели использовался метод стохастизации одношаговых процессов [8–10].

Применяя метод построения одношаговых процессов строим стохастическую модель модуля RED, которая содержит два основных элемента — источник и получатель. В качестве получателя рассматривается очередь. Источник отправляет пакеты, получатель обрабатывает и посылает подтверждение о принятии пакета. Строим модель исходя из предположения, что источник и получатель взаимодействуют по управлению. Таким образом получаем два одномерных уравнения, одно из них описывает окно TCP, а второе — мгновенную длину очереди. Интенсивность отправки пакетов зависит от размера окна.

Подробно построение стохастической модели управляющего модуля управления трафиком типа RED описано в [4] и в приложении (раздел). В статье получена следующая модель:

$$\begin{cases} \mathrm{d}W = \dfrac{1}{T}\mathrm{d}t - \dfrac{W}{2}\mathrm{d}N + \sqrt{\dfrac{1}{T} + \dfrac{W}{2}\dfrac{\mathrm{d}N}{\mathrm{d}t}}\mathrm{d}V^1, \\ \mathrm{d}Q = \left(\dfrac{W}{T} - C\right)\mathrm{d}Q + \sqrt{\dfrac{W}{T} - C}\mathrm{d}V^2, \\ \dfrac{\mathrm{d}\hat{Q}}{\mathrm{d}t} = w_q C(Q - \hat{Q}). \end{cases}$$

Здесь W — среднее значение размера окна, Q — среднее значение мгновенной очереди, $\hat{Q}$ — экспоненциально взвешенное скользящее среднее от среднего мгновенного значения окна со значением параметра w_q, C — интенсивность исходящего потока из очереди, $\mathrm{d}V^i$ — двумерный винеровский случайный процесс.

III. СТРУКТУРА ИМЕНОВАНИЯ ОБРАЗОВ CISCO IOS

Для использования образа IOS следует представлять себе, какие функции поддерживает данный вариант поставки [11]. За это отвечает параметр «Feature set» (схема 1):

- IP Base — начальный уровень функциональности, включается во все другие наборы возможностей. Обеспечивает базовый роутинг, то есть статические маршруты, RIP, OSPF, EIGRP, только на IPv4. Включает VLAN (802.1Q и ISL), которые ранее были доступны только в наборе IP Plus. Также включает NAT.

* trvelieva@gmail.com

† akorolkova@sci.pfu.edu.ru

‡ yamadharma@gmail.com

§ Опубликовано в: *Velieva T. R., Korolkova A. V., Kulyabov D. S.* Designing installations for verification of the model of active queue management discipline RED in the GNS3 // 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). — IEEE, 2014. — P. 570–577. — doi:10.1109/ICUMT.2014.7002164. ; Исходные тексты: `https://bitbucket.org/yamadharma/articles-2014-gns3-red`

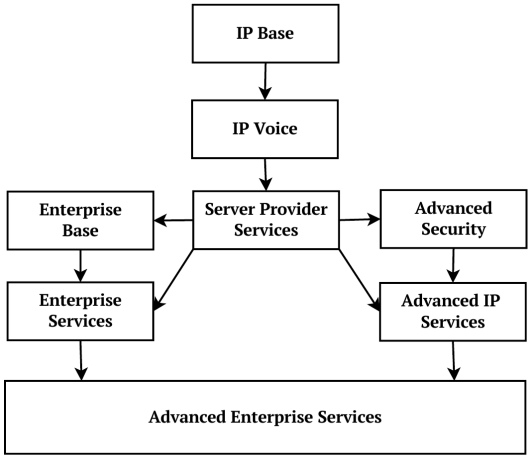

Рис. 1. Дерево свойств Cisco IOS

- IP Services (для коммутаторов 3-го уровня) — протоколы динамической маршрутизации, NAT, IP SLA.
- Advanced IP Services — добавляется поддержка IPv6.
- IP Voice — добавляет функциональность VoIP и VoFR.
- Advanced Security — добавляется IOS/Firewall, IDS, SCTP, SSH и IPSec (DES, 3DES и AES).
- Service Provider Services — добавляется IPv6, Netflow, SSH, BGP, ATM и VoATM.
- Enterprise Base — добавляется поддержка унаследованных L3 протоколов, таких как IPX и AppleTalk. Также включаются IBM features типа DLSw+, STUN/BSTUN и RSRB.

Начиная с версии IOS 12.3T Cisco использует новую схему именования образов (таб. I). Однако данный метод именования не покрывает всех тонкостей комплектования образа, поэтому до сих пор используются элементы старой схемы именования (таб. II).

По невыясненным причинам не все образы IOS работоспособны в GNS3. Нами опробовано несколько образов. Далее мы приводим краткий список работающих и неработающих образов.

Работоспособные образы:

- `C1700-adventerprisek9-mz.124-8.`
- `C1710-bk9no3r2sy-mz.124-23`
- `C1720-l2sy-mz.121-11`
- `C2600-adventerprisek9_sna-mz.124-25b`
- `C2691-adventerprisek9_sna-mz.124-23`
- `C3660-jsx-mz.123-4.T.`

Таблица I. Новая структура именования набора возможностей

Код	Набор возможностей
Base	Entry level image (IP Base, Enterprise Base)
Services	Addition of IP Telephony Service, MPLS, Voice over IP, Voice over Frame Relay and ATM (Included in SP Services, Enterprise Services)
Advanced	Addition of VPN, Cisco IOS Firewall, 3DES encryption, SSH, Cisco IOS IPSec and Intrusion Detection Systems (IDS) (Advanced Security, Advanced IP Services)
Enterprise	Addition of multi-protocols, including IBM, IPX, AppleTalk (Enterprise Base, Enterprise Services)

Таблица II. Старая структура именования набора возможностей

Код	Набор возможностей
I	IP
Y	IP on 1700 Series Platforms
S	IP Plus
S6	IP Plus – No ATM
S7	IP Plus – No Voice
J	Enterprise
O	IOS Firewall/Intrusion Detection
K	Cryptorgaphy/IPSEC/SSH
K8	56Bit DES Encryption (Weak Cryptography)
K9	3DES/AES Encryption (Strong Cryptography)
X	H323
G	Services Selection Gateway (SSG)
C	Remote Access Server or Packet Data Serving Node (PDSN)
B	Apple Talk
N	Novel IP/IPX
V	Vox
R	IBM
U	Unlawful Intercept
P	Service Provider
Telco	Telecommunications Feature Set
Boot	Boot Image (Used on high end routers/switches)

- `C3745-adventerprisek9_sna-mz.124-15.T14.`
- `C7200-adventerprisek9_sna-mz.152-4.m4`

Неработоспособные образы:

- `C2600-adventerprisek9-sna-mz.124-23.`
- `C3745-adventerprisek9_sna-mz.124-15.T14`
- `C3745-adventerprisek9_ivs-mz.124-15.T14`
- `C3745-adventerprisek9_ivs-mz.124-15.T8`

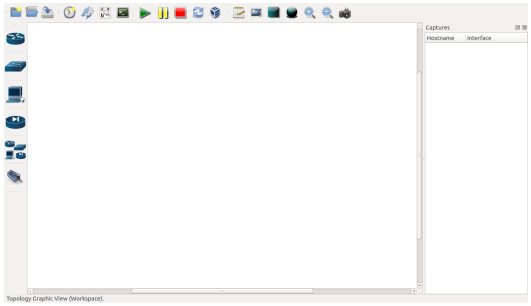

Рис. 2. Интерфейс GNS3

IV. УСТАНОВКА И НАСТРОЙКА GNS3

GNS3 — это комплекс, который позволяет смоделировать виртуальную сеть из маршрутизаторов и виртуальных машин [12]. Работает практически на всех платформах. Фактически это графический интерфейс для разных виртуальных машин. Для эмуляции устройств Cisco используется эмулятор dynamips. Кроме того, можно использовать такие эмуляторы, как VirtualBox и Qemu. Последний особенно удобен при использовании с системой KVM, позволяющей использовать аппаратную реализацию процессора.

Графический интерфейс позволяет легко коммутировать разные виртуальные машины. Кроме того, есть возможность соединения проектируемой топологии с реальной сетью. Использование WireShark позволяет провести мониторинг трафика внутри проектируемой топологии.

Для работы с GNS3 необходимо установить Dynamips, VirtualBox и/или QEMU, xdotool, Wireshark. Чтобы установить вышеуказанное программное обеспечение для операционных систем семейства Linux (в нашем случае GNS3 устанавливался на Ubuntu 14.04) в консоли прописываем следующие команды:

```
sudo apt-get install dynamips
sudo apt-get install qemu
sudo apt-get install virtualbox
sudo apt-get install xdotool
sudo apt-get install wireshark
```

После этого устанавливаем сам GNS3, аналогичным образом прописывая команду в терминале:

```
sudo apt-get install gns3
```

Далее запускаем GNS3. Перед нами открывается интерфейс GNS3 (см. рис. 2).

Вверху расположено контекстное меню, с левой стороны оборудование на выбор, снизу — консольное окно программы, справа — меню управления сетью. Приступая к работе, необходимо провести предварительную настройку GNS3.

Для этого в контекстном меню во вкладке Edit выбираем пункт Preferences (рис. 3).

В подменю General есть возможность изменения языка. Здесь прописывается путь к папкам, в которых

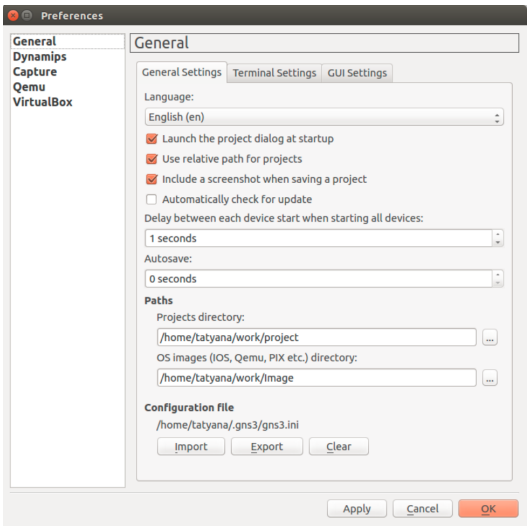

Рис. 3. Общие настройки GNS3

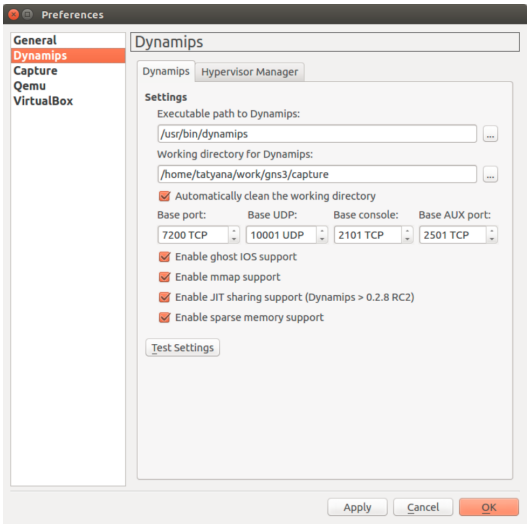

Рис. 4. Настройки эмулятора Dynamips

будут хранится проекты и образы оборудования. В Dynamips прописывается путь к папке, куда установлен dynamips и путь к папке для захвата файлов (рис. 4).

В подменю Capture конфигурируются параметры захвата трафика (рис. 5) (впрочем, на данный момент мы не используем возможность захвата трафика). В строке Default Presents выбираем Wireshark Live Traffic Capture (Linux). В следующей строке указываем путь к директории, в которой будут храниться данные о захвате. И в последней строке прописываем команду для запуска Wireshark и чтения файла захвата:

```
tail -f -c +0b %c | wireshark -k -i
```

В разделе Qemu во вкладке General Settings прописываем путь к Qemuwrapper (этот файл входит в поставку GNS3). Указываем директорию для захвата. В строке Path to qemu прописываем путь к файлу, который

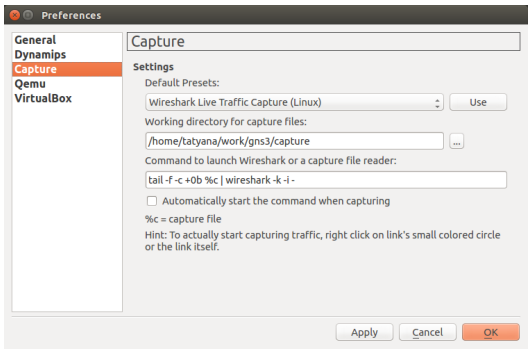

Рис. 5. Настройки захвата трафика

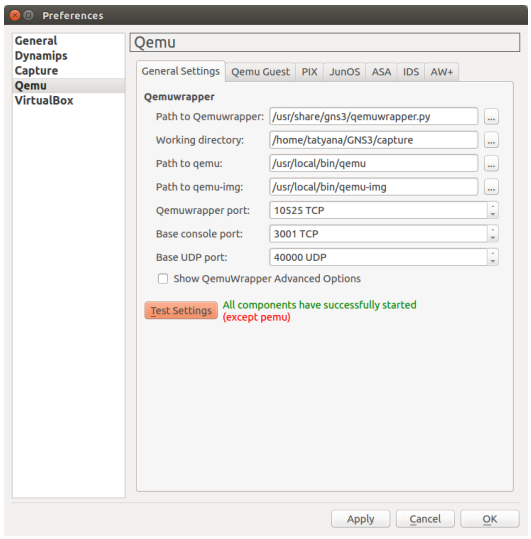

Рис. 6. Preferences GNS3 - Qemu

эмулирует работу процессора. В следующей строке указываем путь к образу виртуальной машины. Номера портов оставляем по умолчанию. Чтобы проверить работу Qemuwrapper нажимаем на кнопку Test Settings, должна появиться зеленая надпись о выполнении теста (рис. 6).

Далее, устанавливаем образ ОС на виртуальную машину. Набор образов можно взять со страницы `http://www.gns3.net/appliances/`. Мы выбрали Linux Core 4.7.7, поскольку он содержит генератор трафика D-ITG.

В строке Qemu flavor выставляем интересующую нас модель процессора, указываем имя. После этого прописываем путь к образу ОС. Нажимаем Save и OK. После этого на панели элементов можно будет выбрать Qemu guest (рис. 7).

Теперь можно собирать стенд. Для этого переносим из меню выбора устройства маршрутизатор и виртуальную машину Qemu guest, переименовываем (правая кнопка мыши » change the hostname) (рис. 8).

На данном стенде, host01 — источник пакетов; host02 — получатель. Настроим слоты для маршрутизатора. Для этого нажимаем правой кнопкой мыши

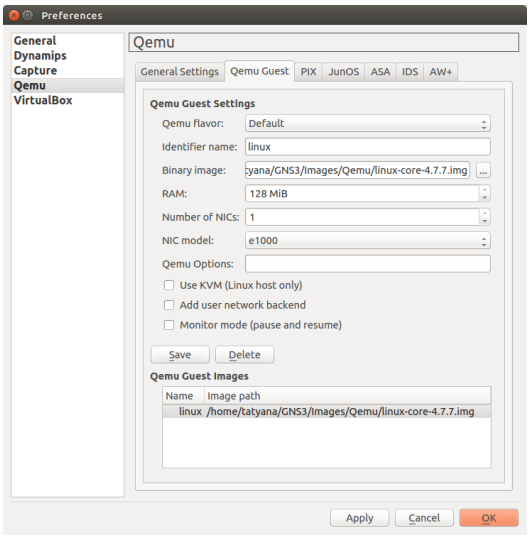

Рис. 7. Preferences GNS3

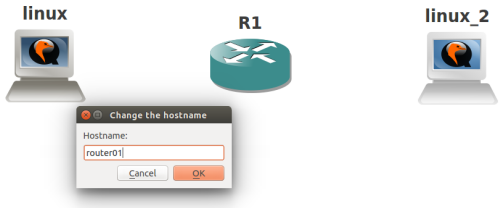

Рис. 8. Virtual Network

на router01 и выбираем Configure » router01 » Slots. Для slot0 выбираем FastEthernet (рис. 9).

Теперь создаем соединение между маршрутизатором и виртуальной машиной. Правой кнопкой мыши на host01 выбираем add link » FastEthernet » e0 и соединяем с router01, выбирая f0/0. Аналогично этому создаём соединение между host02 и router01, только теперь

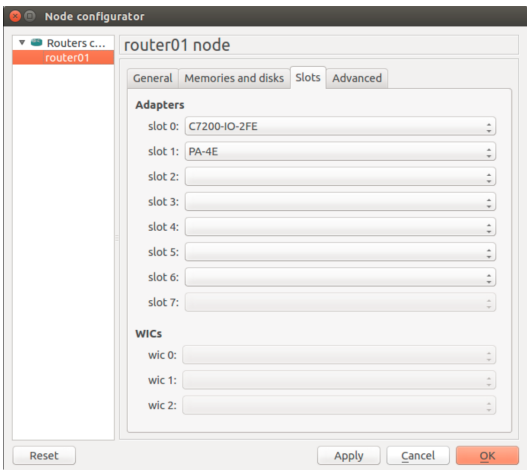

Рис. 9. Router configurator

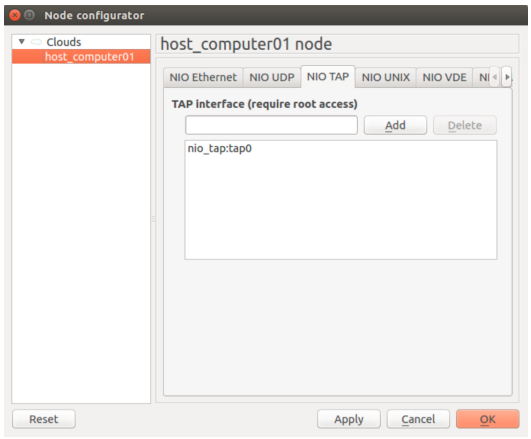

Рис. 10. Cloud interface configurator

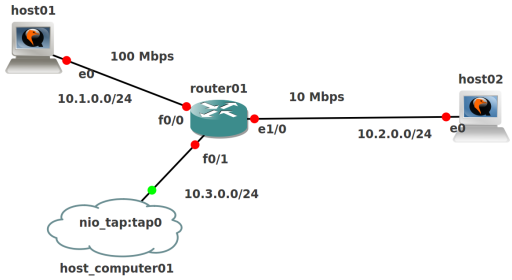

Рис. 11. Virtual Network

выбираем тип соединения Ethernet (скорость 10 Mbps).

Теперь нам нужно соединить виртуальные машины с хостовым компьютером (для получения и обработки логов). Соединение будем осуществлять через TUN/TAP интерфейс (TAP симулирует Ethernet устройство и работает на канальном уровне модели OSI, оперируя кадрами Ethernet и используется для создания моста). Для этого перетаскиваем на рабочую область облако и конфигурируем его: Configure » NIO TAP. Вписываем название tap0 и нажимаем add » apply » ok (рис. 10).

Создаём соединение с router01, выбирая тип соединения FastEthernet. Получаем готовую топологию (рис. 11).

Далее необходимо произвести конфигурацию устройств.

Запускаем router01 (правая кнопка » start). Открываем консоль для него (правая кнопка » Console) и вводим следующие команды (рис. 12):

```
Router>enable
Router#configure terminal
router01(config)#hostname router01
router01(config)#interface f0/0
router01(config-if)#ip address 10.1.0.1
  255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#interface e1/0
```

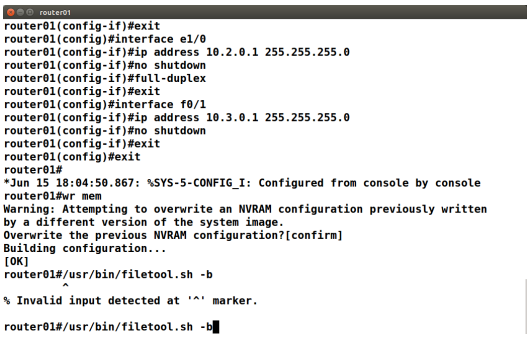

Рис. 12. Router CLI configuration

```
router01(config-if)#ip address 10.2.0.1
  255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#full-duplex
router01(config-if)#exit
router01(config)#interface f0/1
router01(config-if)#ip address 10.3.0.1
  255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#exit
router01#write memory
```

Теперь открываем консоль для host01. В качестве имени пользователя задаём `tc`, пароль пустой. Переходим в режим суперпользователя: `sudo su`. Редактируем параметры host01 (для начала редактирования в редакторе vi вводим `i`; чтобы сохранить изменения и выйти из редактора нажимаем кнопку esc, затем вводим `:wq` и нажимаете кнопку Enter). Придётся отредактировать два файла: `/opt/bootsync.sh` и `/opt/bootlocal.sh`:

```
vi /opt/bootlocal.sh
```

В конце файла прописываем следующее:

```
ifconfig eth0 10.1.0.10 netmask 255.255.255.0 up
route add default gw 10.1.0.1
```

После сохранения изменений в файле выходим из него и выполняем команду для сохранения конфигурации (Linux Core использует систему конфигурации Tinycore):

```
/usr/bin/filetool.sh -b
```

Аналогично конфигурируется host02, изменяются только ip адреса (сеть 10.2.0.0/24).

Теперь сконфигурим хостовый компьютер, подняв на нём TAP-интерфейс. В командном окне в режиме суперпользователя создаем интерфейс для пользователя:

```
tunctl -u имя_пользователя -t tap0
```

Устанавливаем адрес (рис. 13):

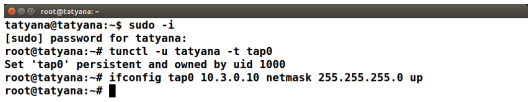

Рис. 13. Address configuration on tap0 interface

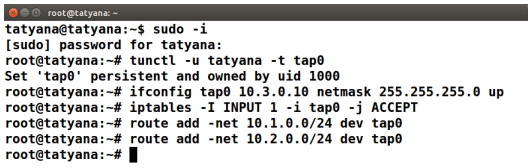

Рис. 14. Routing configuration on tap0 interface

```
ifconfig tap0 10.3.0.10 netmask 255.255.255.0 up
```

Далее производятся настройка фильтрации и маршрутизация (рис. 14):

```
iptables -I INPUT 1 -i tap0 -j ACCEPT
route add -net 10.1.0.0/24 dev tap0
route add -net 10.2.0.0/24 dev tap0
```

V. ГЕНЕРАТОР ТРАФИКА D-ITG

Теперь можно генерировать трафик и снимать показания. Для генерации трафика мы используем D-ITG.

D-ITG позволяет получить оценки основных показателей качества обслуживания (средняя задержка передачи пакетов, вариация задержки (джиттер), коэффициент потерь пакетов, производительность) с высокой степенью достоверности. В зависимости от требований в ходе проведения лабораторного эксперимента можно изменять следующие параметры:

- количество потоков, передаваемых между конечными станциями;
- продолжительность генерирования трафика;
- интенсивность каждого отдельного потока (пак/с или бит/с);
- длина пакетов трафика или их закон распределения;
- вид и параметры закона распределения временного интервала между соседними пакетами;
- тип протокола транспортного уровня: TCP, UDP, DCCP, SCTP;
- тип трафика (имитация потока, создаваемого определенным протоколом прикладного уровня): VoIP, IPTV, Telnet, DNS, игровой трафик (Counter Strike, Quake3).

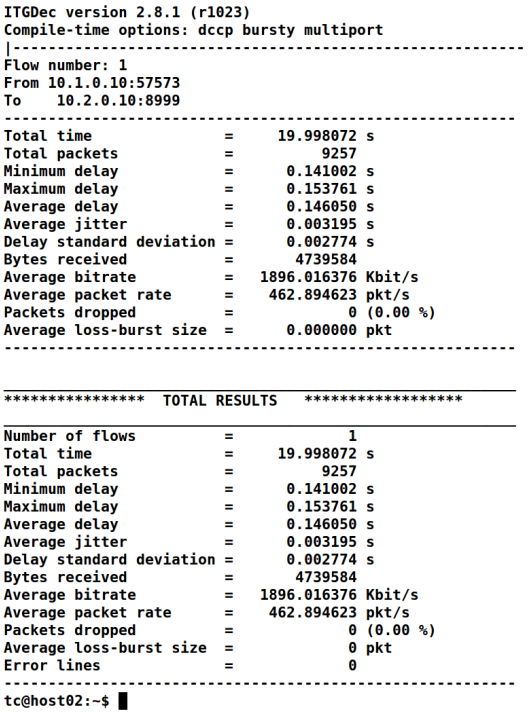

Рис. 15. Статистика входного потока. UDP

В рамках D-ITG управление экспериментом осуществляется при помощи командной строки, и необходимый набор параметров для генерирования трафика задается путем вызова программы `ITGSend` с использованием ключей.

Рассмотрим пример генерации трафика по протоколам UDP и TCP. В консоли host02 выполняем команду на прослушивание канала:

```
ITGRecv
```

В консоли host01 запустим генерацию трафика:

```
ITGSend -a 10.2.0.10 -T UDP -C 10000 -c 500 \
  -t 20000 -x recv_log_file
```

Здесь мы передаем файлы получателю с адресом 10.2.0.10 по протоколу UDP, скорость передачи 10000 пакетов в секунду, размер пакета 500 байт, время соединения 20000 мс. Вся информация по отправке данных записывается на стороне получателя в файл с названием `recv_log_files`. После того как передача данных была выполнена, останавливаем прослушивание в консоли host02 (нажимая Ctrl + C), затем в консоли host02 выполняем команду на декодирование лог-файла:

```
ITGDec recv_log_file
```

При помощи данной команды на экран выводится таблица со значениями параметров принятого потока (рис. 15).

Аналогично выполняем для TCP и получаем следующее (рис. 16).

```
ITGDec version 2.8.1 (r1023)
Compile-time options: dccp bursty multiport
\------------------------------------------------------------
Flow number: 1
From 10.1.0.10:34185
To    10.2.0.10:8999
------------------------------------------------------------
Total time                =     20.015369 s
Total packets             =          4742
Minimum delay             =      0.132785 s
Maximum delay             =      1.180837 s
Average delay             =      0.190162 s
Average jitter            =      0.005034 s
Delay standard deviation  =      0.137616 s
Bytes received            =       2371000
Average bitrate           =    947.671762 Kbit/s
Average packet rate       =    236.917940 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS    ******************

Number of flows           =             1
Total time                =     20.015369 s
Total packets             =          4742
Minimum delay             =      0.132785 s
Maximum delay             =      1.180837 s
Average delay             =      0.190162 s
Average jitter            =      0.005034 s
Delay standard deviation  =      0.137616 s
Bytes received            =       2371000
Average bitrate           =    947.671762 Kbit/s
Average packet rate       =    236.917940 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
root@host02:~#
```

Рис. 16. Статистика входного потока. TCP

Для многопотоковой передачи трафика в командном окне host01 в файле `send` построчно описываем каждый поток. Например, таким образом:

```
cat > send <<EOF
-a 10.2.0.10  -C 1000 -c 512 -T UDP
-a 10.2.0.10  -C 2000 -c 512 -T UDP
-a 10.2.0.10  -C 3000 -c 512 -T UDP
-a 10.2.0.10  -C 4000 -c 512 -T UDP
-a 10.2.0.10  -C 5000 -c 512 -T UDP
EOF
```

Теперь запускаем передачу трафика на host01. Данные о передаче трафика с host01 на host02 записываются в файлы send.log и recv.log соответственно:

```
ITGSend send -l send.log -x recv.log
```

Выведем на экран отчет о передаче потоков. Для этого в консоли host02 напишем команду

```
ITGDec send.log
```

для дешифровки отправляемого трафика (рис. 17) и команду

```
ITGDec recv.log
```

для дешифровки получаемого трафика.

Таким образом получаем данные о каждом потоке, а также значение параметров исходящего и входящего потоков.

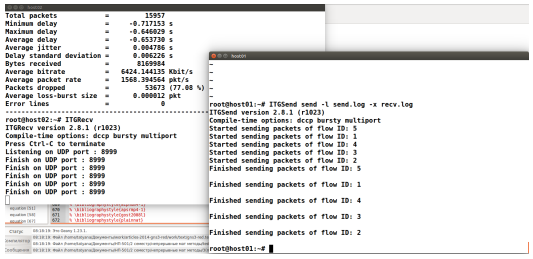

Рис. 17. Дешифровка лога

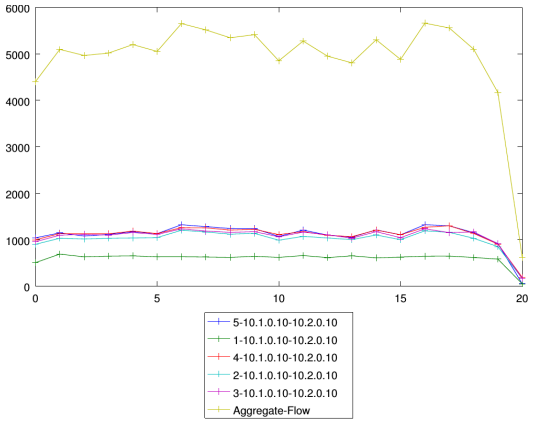

Рис. 18. Потеря пакетов (UDP)

VI. ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ

По результатам эксперимента можно построить графики для разных характеристик, как то:

- задержка передачи пакетов (delay);
- вариация задержки (jitter);
- интенсивность (bitrate);
- коэффициент потерь пакетов (packet loss).

Рассмотрим пример построения графиков. Предположим, что у нас есть 5 потоков, которые передаются по протоколу TCP. Скорость передачи 1 потока — 1000 пакетов в секунду, 2 потока — 2000 пакетов в секунду, 3 потока — 3000 пакетов в секунду, 4 потока — 4000 пакетов в секунду, 5 потока — 5000 пакетов в секунду. Размеры пакетов у всех потоков одинаковы — 512 байт, а время соединения 20000 мс. Для демонстрации потери пакетов будем использовать протокол UDP (рис. 18).

При помощи `ITGplot` построим графики для bitrate, delay и jitter. Для этого нужно записать полученные значения в отдельные файлы при помощи следующих команд:

```
ITGDec recv.log -b 1000
ITGDec recv.log -d 1000
ITGDec recv.log -j 1000
```

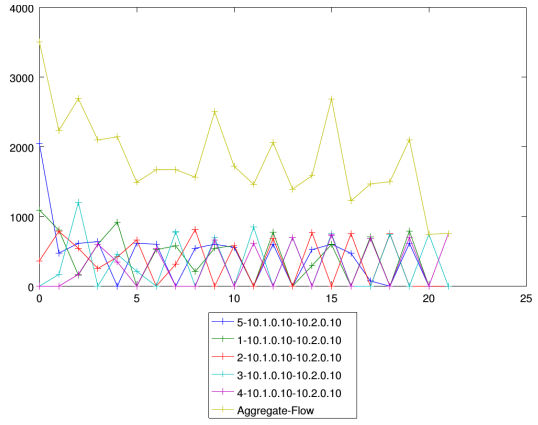

Рис. 19. Интенсивность поступающего потока

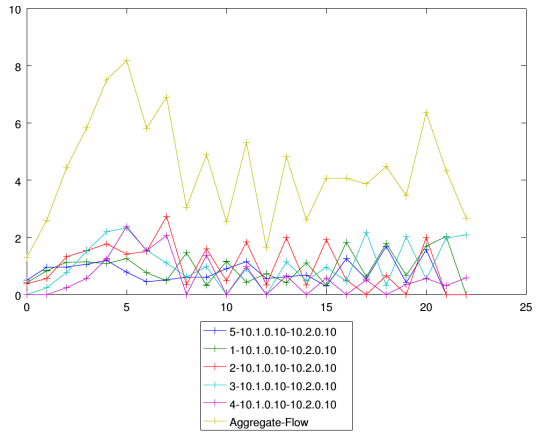

Рис. 20. Задержка получения пакетов

Здесь через каждые 1000 мс в файлы bitrate.dat, jitter.dat, delay.dat записываются соответственные значения параметров передачи для потоков (рис. 19, 20, 21). Построим графики при помощи следующих команд:

```
./ITGplot birate.dat
./ITGplot delay.dat
./ITGplot jitter.dat
```

На полученных графиках верхняя кривая отображает общее поведение всех потоков.

VII. ВЫВОДЫ

Таким образом, мы построили основу стенда для верификации модели модуля управления трафиком RED. Однако, в данной работе не рассмотрена настройка самого модуля RED на маршрутизаторе и принципы снятия показаний с роутера в реальном времени.

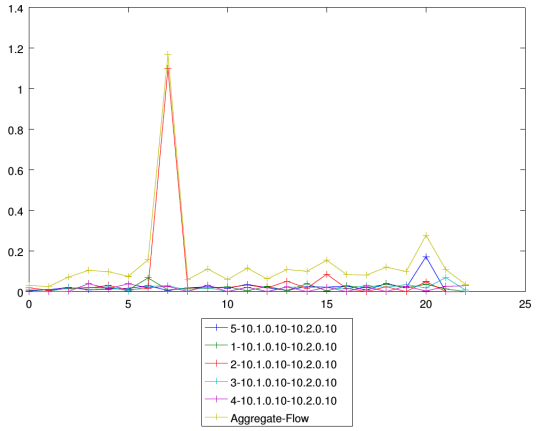

Рис. 21. Неравномерность получения пакетов

Приложение: Построение стохастической модели модуля управления трафиком RED

Изменение окна при каждом подтверждении (ACK)

$$W_{n+1} = W_n + \frac{1}{W_n}.$$

Для применения метода стохастизации одношаговых процессов мы должны перейти к модели с непрерывным временем. Непрерывное уравнение для окна T — время двойного оборота. За это время приходят W_n подтверждений.

$$\dot{W} = \frac{1}{T}.$$

Дискретное уравнение мгновенной длины очереди:

$$Q_{n+1} = Q_n + W_n - C_n.$$

На основе дискретного уравнения мгновенной длины очереди выводится непрерывное уравнение, которое записывается следующим образом:

$$\dot{Q} = \frac{W(t)}{T(t)} - C(t).$$

Далее опишем поведение экспоненциально взвешенной скользящей средней длины очереди, которая представляет собой уравнение связи между источником и получателем.

Дискретное реккурентное уравнение средней длины очереди

$$\hat{Q}(t_k + \delta) = (1 - w_q)\hat{Q}(t_k) + w_q Q(t_k).$$

Из дискретного уравнение выводим непрерывное уравнение средней длины очереди

$$\dot{\hat{Q}} = -\frac{w_q}{\delta}\hat{Q} + \frac{w_q}{\delta}Q.$$

Чтобы рассмотреть поведение пакетов в системе строим кинетическое уравнение или как его еще называют уравнение взаимодействия. Количество пакетов задаётся окном TCP.

$$\begin{cases} 0 \xrightarrow{k_1^1} W, \\ W \xrightarrow{k_2^1} 0. \end{cases}$$

Первое соотношение описывает появление пакетов в системе, второе — вывод пакетов из системы.

На основании записанных уравнений, применяя метод построения одношаговых процессов, выводим уравнение Фоккера–Планка:

$$\frac{\partial w}{\partial t} = -\frac{\partial}{\partial W}\left[\left(\frac{1}{W} - \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}\right)w\right] + \\ + \frac{1}{2}\frac{\partial^2}{\partial W^2}\left[\left(\frac{1}{W} + \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}\right)w\right].$$

Из уравнения Фоккера–Планка получаем соответствующее ему уравнение Ланжевена:

$$\mathrm{d}W = \frac{1}{W}\mathrm{d}t - \frac{W}{2}\mathrm{d}N + \sqrt{\frac{1}{W} + \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}}\mathrm{d}V^1,$$

где $\mathrm{d}V^1$ — винеровский процесс, соответствующий случайному процессу $W(t)$.

Аналогичным образом описывается поведение длины очереди. Уравнения взаимодействия для мгновенной длины очереди:

$$\begin{cases} 0 \xrightarrow{k_1^2} Q \\ 0 \xrightarrow{k_2^2} Q. \end{cases}$$

Уравнение Фоккера–Планка для мгновенной длины очереди:

$$\frac{\partial q}{\partial t} = -\frac{\partial}{\partial Q}\left[\left(\frac{W}{T} - C\right)q\right] + \frac{1}{2}\frac{\partial^2}{\partial Q^2}\left[\left(\frac{W}{T} - C\right)q\right].$$

Уравнение Ланжевена для мгновенной длины очереди:

$$\mathrm{d}Q = \left(\frac{W}{T} - C\right)\mathrm{d}t + \sqrt{\frac{W}{T} - C}\mathrm{d}V^2,$$

где $\mathrm{d}V^2$ — винеровский процесс, соответствующий случайному процессу Q.

Согласно полученным уравнениям записываем результирующую систему:

$$\begin{cases} \mathrm{d}W = \frac{1}{T}\mathrm{d}t - \frac{W}{2}\mathrm{d}N + \sqrt{\frac{1}{T} + \frac{W}{2}\frac{\mathrm{d}N}{\mathrm{d}t}}\mathrm{d}V^1, \\ \mathrm{d}Q = \left(\frac{W}{T} - C\right)\mathrm{d}Q + \sqrt{\frac{W}{T} - C}\mathrm{d}V^2, \\ \frac{\mathrm{d}\hat{Q}}{\mathrm{d}t} = w_q C(Q - \hat{Q}). \end{cases}$$

Наиболее подробно стохастическая модель управляющего модуля маршрутизатора типа RED описана в [4].

[1] Королькова А. В. Метод расчёта вероятности сброса пакетов в алгоритме RED // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2007. — № 1–2. — С. 32–37.

[2] Королькова А. В., Кулябов Д. С., Черноиванов А. И. К вопросу о классификации алгоритмов RED // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2009. — № 3. — С. 34–46.

[3] Королькова А. В., Кулябов Д. С. Математическая модель динамики поведения параметров систем типа RED // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2010. — № 1. — С. 68–76.

[4] Велиева Т. Р., Королькова А. В., Кулябов Д. С., Сантуш Б. А. Модель управления очередями на маршрутизаторах // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2014. — Т. 2. — С. 81–92.

[5] Floyd S., Jacobson V. Random Early Detection Gateways for Congestion Avoidance // IEEE/ACM Transactions on Networking. — 1993. — Vol. 1, no. 4. — P. 397–413.

[6] Misra V., Gong W.-B., Towsley D. Stochastic Differential Equation Modeling and Analysis of TCP-windowsize Behavior // Proceedings of IFIP WG 7.3 Performance. — 1999. — Vol. 99. — URL: http://dna-pubs.cs.columbia.edu/citation/paperfile/24/Misra99-TCP-Stochastic.pdf.

[7] Misra V., Gong W.-B., Towsley D. Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED // ACM SIGCOMM Computer Communication Review. — 2000. — Vol. 30, no. 4. — P. 151–160.

[8] Кулябов Д. С., Демидова А. В. Введение согласованного стохастического члена в уравнение модели роста популяций // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2012. — № 3. — С. 69–78.

[9] Демидова А. В., Геворкян М. Н., Егоров А. Д. и др. Влияние стохастизации на одношаговые модели // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2014. — № 1. — С. 71–85.

[10] Eferina E. G., Korolkova A. V., Gevorkyan M. N. et al. One-Step Stochastic Processes Simulation Software Package // Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics". — 2014. — no. 3. — P. 46–59.

[11] Cisco Feature Navigator. — URL: www.cisco.com/go/cfn (online; accessed: 2014-07-29).

[12] Welsh C. GNS3 network simulation guide. — PACKT Publisher, 2013. — P. 154. — ISBN: 1782160809. — URL: http://cds.cern.ch/record/1633716.